How is authenthicity and confidentiality maintained for MAM channels on the IOTA Tangle?

Marius Ødegård Lindvall
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology
Gjøvik, Norway
E-mail: mariuoli@stud.ntnu.no

***Abstract*—IOTA is a distributed network that, among other usage areas, allows anyone to create channels of encrypted data that third parties can subscribe to and receive updates from in real-time, through a protocol layer called "masked authentication messaging." This paper will explain how authenthicity and confidentiality is maintained for the messages posted through such channels.**

I. Introduction

IOTA is a distributed ledger technology (DLT) based on ternary logic[1] that, contrary to many other technologies in the field, such as Bitcoin and Ethereum, uses a directed acyclic graph technology called the "Tangle" rather than a blockchain to attach and propagate transactions and data [1]. IOTA allows transferring value in transactions using the IOTA cryptocurrency. However, unlike most other DLTs, IOTA allows zero-value transactions, i.e. transactions which carry only data and no value [1].

This creates several opportunities for decentralized data transfer, especially for sensors and other Internet of Things (IoT) devices [2]. One of the possibilities this opens is the ability for interested parties to subscribe to the data stream from a sensor and receive updates in real-time on, for example, the temperature and humidity of a particular environment, or the air quality index and greenhouse gas emission levels at a popular traffic intersection, without needing servers and infrastructure to facilitate the distribution of that data. A protocol has been designed for this kind of data transfer, called masked authentication messaging, or MAM for short [3].

Masked authentication messaging allows a device connected to a Tangle node to broadcast encrypted messages to a "channel" that anyone with the right access credentials can access and follow. Channels can be forked into multiple data streams, and offer several ways to restrict their contents only to those who have been authorized to view them. Each message in a channel is sent as a zero-value transaction to a particular unique address, the address itself being chosen separately depending on the type of channel (public, private and restricted), and which is only used once. Each message has a reference to address that will contain the next message in the channel, allowing subscribers with access to one message to also find future messages in the data stream [3].

[1]The ternary equivalent of bits and bytes are trits and trytes. One tryte consists of three trits, each of which may have values -1, 0 or 1. The 27 possible combinations of trits that can make up a tryte are represented in IOTA as the number 9, followed by uppercase letters A through Z.

II. Message authenthicity

A core aspect of securing a data stream is to ensure that third parties cannot impersonate its author. MAM channels use their constituent transactions' addresses to identify messages belonging to a channel. However, IOTA is a decentralized network where any node can send any message to any address. This means that if a message in a MAM channel points to a particular address as the location of the next message in the channel, that address is not guaranteed to contain only the legitimate transaction that is part of the MAM channel - it could also contain other transactions that malicious third parties have intentionally sent to try and hijack the channel. This attack vector is mitigated through message signing.

While many DLTs use RSA or similar algorithms for transaction signing, IOTA has opted for using a signature scheme based on Winternitz one time signatures [4] using Kerl, a trinary wrapper for the Keccak hash function, as its hash function [5]. Winternitz signatures have the advantages that they are fast to compute, and offer much greater resilience against quantum computing attacks than traditional factoring algorithms such as RSA [6]–[8], but come with the drawback that each signature can only be used once—re-using a signing key makes it significantly easier for threat actors to forge an authenticated signature for a message, as 50% of the key used to sign the message is exposed with every signature [5], [9].

MAM transactions are signed through Merkle trees. The root of the Merkle tree is used as the ID of the MAM channel, and a new Merkle tree is generated for each message in the channel [3]. The leaves of the Merkle trees are hashes generated from a combination of the seed, i.e. a private key, chosen by and only known to the MAM channel publisher, combined with the index number of each leaf in the generated tree, starting from $i = 0$ for the first leaf of the first tree of the MAM channel, which together form a subseed [10], [11]. The publisher, who has the seed, can generate any Merkle tree starting at any index and with any depth/number of leaves for that specific seed. Because the Merkle trees for future messages use leaves hashed from the same seed as the previous message, where the index of the first leaf of the tree is the next index from that of the last leaf of the last tree, the publisher

of the channel can generate Merkle trees for all successive messages in their channel at any point in time [10].

The channel publisher signs their message using one of the leaves of the Merkle tree, and attaches the signature of the message, the index of the used leaf in the tree, its siblings, and the root of the Merkle tree that will be used to sign the next message in the channel, to the MAM message [3], [10]. They can then send the channel ID to parties who want to subscribe to the channel, who will then look up the message in the Tangle by its transaction address, found using the channel ID, i.e. Merkle tree root [10].

When a party wants to validate a message in the channel, they validate the signature of the transaction and match it with the attached siblings to form a Merkle tree root [12]. If that root matches the provided channel ID, the message is verified, and is thus known to originate from the correct MAM channel publisher. The message also provides the Merkle root of the next message in the channel, allowing subscribers to look up the next message on the Tangle and repeat the process using the next root as the channel ID for future messages [3], [10].

III. Channel confidentiality

All MAM messages are, along with the Merkle tree signing leaf index, its siblings, and the root of the next Merkle tree, encrypted [10] using a ternary sum operation [13] that behaves similarly to the binary XOR, in practice performing a modulo 3 on the sum of two trits [14]. Encryption of the payload is done in blocks of 243 trits—for the first block, an optional side key (explained later) is absorbed by the Kerl hash function, followed by the encryption key, the leaf index, and the length of the encoded message [12]. After this operation, the hash function will have a certain state—an array of trits that varies based on the absorbed input. The first 243 trits of that state is then used as the key that is summed into the payload block [13]. The unencrypted block is absorbed into the same hash function instance, in turn causing the state of the hash function to update. The 243 first trits of the new state are now used to sum the next block of 243 trits from the payload. This repeats until the entire block has been encrypted [13].

When MAM messages are published on the Tangle, they are sent to a certain target address. MAM supports three privacy/encryption modes for MAM payloads—public, private, and restricted—that change how the target address is generated and encryption is performed [3], [10]. In public mode, the Merkle root, i.e. channel ID, is used as the address of the transaction, as well as the encryption key. This means that anyone can stumble across a MAM transaction, use its address to decrypt the payload, read the data, and find the address of the next message in the channel. Private mode uses the *hash* of the root as the transaction address instead—anyone can still find the transaction, but since the hash is not reversible, the root cannot be found to decrypt the payload, meaning the root must be known to read the message. Restricted mode works like private mode, but requires an additional side key as the encryption key of the payload, hence both the side key and root must be known to read data from the MAM channel [10].

IV. Conclusion

When data is exchanged between two parties, it is vital that third parties cannot impersonate one of the parties and provide falsified data to the other party. Depending on the type of data that is exchanged, it may also be necessary to ensure that the transferred data is kept confidential, and not accessible to uninvited third parties. Thus, the need for data authentication and confidentiality arises. IOTA, being a distributed ledger, principally allows anyone to make transactions, though through usage of a Merkle tree-based signature scheme, a guarantee can be made that data transferred through a MAM channel did in fact originate from a trusted party whose messages on the Tangle can be authenticated and verified. Furthermore, IOTA allows for MAM channel messages to be encrypted using hash-based ternary sum symmetric encryption, preventing unauthorized parties from listening to the data stream—thus, both authenthicity and confidentiality of data can be preserved for MAM data streams on the IOTA Tangle.

References

[1] S. Popov, "The Tangle," *IOTA Foundation*, 01-Oct-2017. [Online]. Available: https://iotatoken.com/IOTA_Whitepaper.pdf. [Accessed: 12-Oct-2018].

[2] V. A. Red, "Practical comparison of distributed ledger technologies for IoT," *Disruptive Technologies in Sensors and Sensor Systems*, Apr. 2017.

[3] P. Handy, "Introducing Masked Authenticated Messaging," *IOTA Foundation*, 04-Nov-2017. [Online]. Available: https://blog.iota.org/introducing-masked-authenticated-messaging-e55c1822d50e. [Accessed: 13-Oct-2018].

[4] E. Heilman, N. Narula, G. Tanzer, J. Lovejoy, M. Colavita, M. Virza, and T. Dryja, "Cryptanalysis of Curl-P and Other Attacks on the IOTA Cryptocurrency." [Online]. Available: https://data.hackinn.com/ppt/BlackHat-USA-2018/us-18-Narula-Heilman-Cryptanalysis-of-Curl-P-wp.pdf. [Accessed: 14-Oct-2018].

[5] E. Hop, "Exploring the IOTA signing process," *IOTA Demystified*, 17-Mar-2018. [Online]. Available: https://medium.com/iota-demystified/exploring-the-iota-signing-process-eb142c839d7f. [Accessed: 14-Oct-2018].

[6] M. Green, "Hash-based Signatures: An illustrated Primer," *A Few Thoughts on Cryptographic Engineering*, 18-Apr-2018. [Online]. Available: https://blog.cryptographyengineering.com/2018/04/07/hash-based-signatures-an-illustrated-primer/. [Accessed: 14-Oct-2018].

[7] A. Langley, "Hash based signatures," *ImperialViolet*, 18-Jul-2013. [Online]. Available: https://www.imperialviolet.org/2013/07/18/hashsig.html. [Accessed: 14-Oct-2018].

[8] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[9] "Very basic estimates on the risk of reusing the Winternitz signatures," 01-Sep-2017. [Online]. Available: https://public.tangle.works/winternitz.pdf. [Accessed: 14-Oct-2018].

[10] ABmushi, "IOTA: MAM Eloquently Explained," *Coinmonks*, 24-Feb-2018. [Online]. Available: https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413. [Accessed: 14-Oct-2018].

[11] IOTA, "iota.rs/merkle/src/simple.rs," *GitHub*. [Online]. Available: https://github.com/iotaledger/iota.rs/blob/master/merkle/src/simple.rs. [Accessed: 14-Oct-2018].

[12] IOTA, "MAM/mam/src/mam.rs," *GitHub*. [Online]. Available: https://github.com/iotaledger/MAM/blob/master/mam/src/mam.rs. [Accessed: 14-Oct-2018].

[13] IOTA, "MAM/mam/src/mask.rs," *GitHub*. [Online]. Available: https://github.com/iotaledger/MAM/blob/master/mam/src/mask.rs. [Accessed: 14-Oct-2018].

[14] IOTA, "trit_sum() in iota.rs/tmath/src/add.rs," *GitHub*. [Online]. Available: https://github.com/iotaledger/iota.rs/blob/master/tmath/src/add.rs#L62. [Accessed: 14-Oct-2018].